Университет ИТМО

**Отчет**

По лабораторной работе №2

Метод решения СЛАУ “Метод Гаусса с выбором главного элемента”

Выполнил:

Нодири Хисравхон

P3231

Преподаватель:

Перл Ольга Вячеславовна

Санкт-Петербург

2024

Оглавление

[Задача 3](#_Toc162381748)

[Описание метода 3](#_Toc162381749)

[Блок схема алгоритма: 5](#_Toc162381750)

[Код программы 6](#_Toc162381751)

[Примеры работ 7](#_Toc162381752)

[Вывод 8](#_Toc162381753)

# Задача

Решите систему линейных алгебраических уравнений, реализуя метод Гаусса с выбором главного элемента. Также рассчитайте невязки.

Формат входных данных:

n

a11 a12 ... a1n b1

a21 a22 ... a2n b2

...

an1 an2 ... ann bn

Формат вывода:

x1

x2

...

xn

r1

r2

...

rn

, где x1..xn - значения неизвестных, а r1..rn - невязки.

Для систем, которые не имеют решений или имеют неограниченное количество решений, должно быть напечатано только следующее сообщение:

"The system has no roots of equations or has an infinite set of them.". Для этого задайте значение переменной isSolutionExists и сообщение об ошибке.

## Описание метода

Метод Гаусса с выбором главного элемента — это эффективный алгоритм для решения систем линейных алгебраических уравнений (СЛАУ). Он является модификацией традиционного метода Гаусса, цель которой — увеличить численную стабильность алгоритма за счёт минимизации ошибок округления. Это достигается путём выбора на каждом шаге преобразования системы такого элемента матрицы коэффициентов, который имеет максимальное по модулю значение, для использования в качестве опорного (ведущего).

Основные этапы метода:

1. Выбор главного элемента. На *k*-ом шаге преобразования исходной или промежуточной матрицы системы из всех элементов *aij*, где *i* ≥ *k* и *j* = *k*, выбирается элемент с максимальным абсолютным значением:

*apk* =max|*aik*|

*i*≥*k*

Здесь *apk* — выбранный главный элемент, *p* — номер строки, где он находится. Далее строка *p* меняется местами с текущей строкой *k*, если *p* ̸= *k*.

1. Прямой ход. Используя выбранный главный элемент, проводится исключение неизвестных, то есть преобразование системы таким образом, чтобы ниже элемента *akk* в столбце *k* стояли нули. Для этого каждая строка *i > k* матрицы умножается на *akk* и из неё вычитается строка *k*, умноженная на *aik*:

Изображение выглядит как черный, темнота

Автоматически созданное описание

Аналогично преобразуется и вектор свободных членов *bi*.

1. Обратный ход. После приведения системы к верхнетреугольному виду решение находится методом обратной подстановки. ЕслиИзображение выглядит как черный, темнота

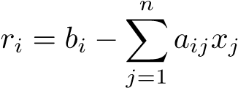
   Автоматически созданное описание, то для всех *i < n* можно найти:

Изображение выглядит как черный, темнота

Автоматически созданное описание

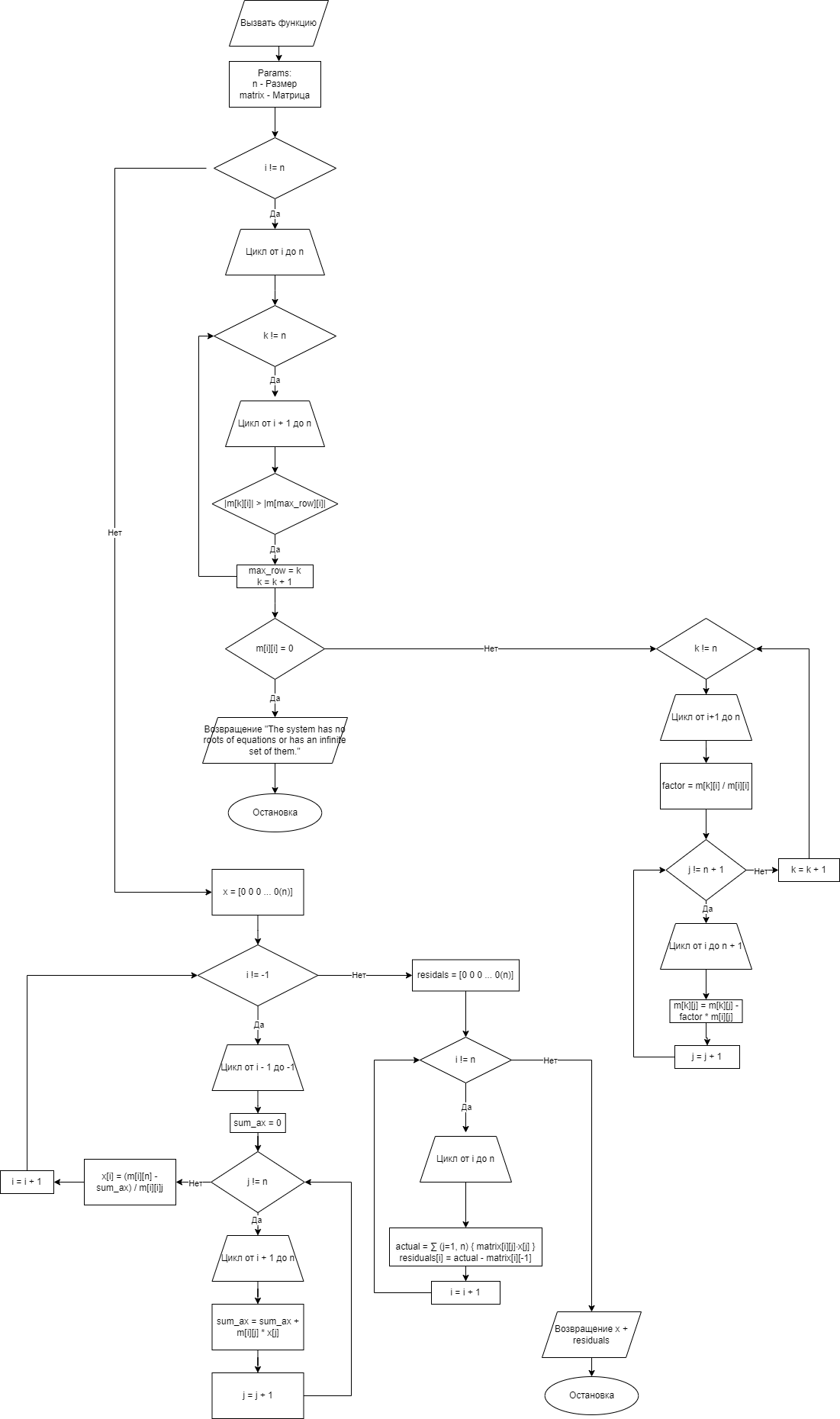
Вычисление невязок:

После нахождения решения полезно вычислить невязки, чтобы оценить точность решения. Невязка для *i*-го уравнения определяется как:



где *xj* — найденные значения неизвестных невязки позволяют оценить, насколько точно найденные решения удовлетворяют исходной системе уравнений. Этот метод особенно ценится за его способность эффективно справляться с численными ошибками, что делает его предпочтительным выбором для решения сложных СЛАУ, особенно в условиях ограниченной точности вычислительной техники.

# Блок схема алгоритма:



## Код программы

#!/bin/python3

**import** **math**

**import** **os**

**import** **random**

**import** **re**

**import** **sys**

**class** **Solution**:

isSolutionExists = True

errorMessage = ""

**@staticmethod**

**def** **solveByGauss**(n, matrix):

**for** i **in** range(n):

max\_row = i

**for** k **in** range(i+**1**, n):

**if** abs(matrix[k][i]) > abs(matrix[max\_row][i]):

max\_row = k

matrix[i], matrix[max\_row] = matrix[max\_row], matrix[i]

**if** matrix[i][i] == **0**:

Solution.isSolutionExists = False

Solution.errorMessage = "The system has no roots of equations or has an infinite set of them."

**return** []

**for** k **in** range(i+**1**, n):

factor = matrix[k][i] / matrix[i][i]

**for** j **in** range(i, n+**1**):

matrix[k][j] -= factor \* matrix[i][j]

x = [**0** **for** \_ **in** range(n)]

**for** i **in** range(n-**1**, -**1**, -**1**):

sum\_ax = **0**

**for** j **in** range(i+**1**, n):

sum\_ax += matrix[i][j] \* x[j]

x[i] = (matrix[i][n] - sum\_ax) / matrix[i][i]

residuals = [**0** **for** \_ **in** range(n)]

**for** i **in** range(n):

actual = sum(matrix[i][j] \* x[j] **for** j **in** range(n))

residuals[i] = actual - matrix[i][-**1**]

**return** x + residuals

**if** \_\_name\_\_ == '\_\_main\_\_':

n = int(input().strip())

matrix\_rows = n

matrix\_columns = n + **1**

matrix = []

**for** \_ **in** range(matrix\_rows):

matrix.append(list(map(float, input().rstrip().split())))

result = Solution.solveByGauss(n, matrix)

**if** Solution.isSolutionExists:

**print**('**\n**'.join(map(str, result)))

**else**:

**print**(f"{Solution.errorMessage}")

# Примеры работ

Пример 1: Уникальное решение

Входные данные:

3

2 1 -1 8

-3 -1 2 -11

-2 1 2 -3

Выходные данные:

2.0

3.0

-1.0

0.0

0.0

0.0

1

Пример 2: Уникальное решение

Входные данные:

2

1 2 9

3 4 24

Выходные данные:

6.0

1.5

0.0

0.0

Пример 3: Нет решения

Входные данные:

2

1 2

2 4

Выходные данные:

The system has no roots of equations or has an infinite set of them.

Пример 4: Бесконечное количество решений

Входные данные:

2

1 2

2 4

Выходные данные:

The system has no roots of equations or has an infinite set of them.

Пример 5: Уникальное решение

Входные данные:

3

10 -7 0 7

-3 2.099 6 3.901

5 -1 5 -6

Выходные данные:

1.0

-2.0

1.0

0.0

0.0

0.0

## Вывод

Метод Гаусса с выбором главного элемента показывает отличную эффективность и точность для решения СЛАУ, значительно повышая численную стабильность и уменьшая риск ошибок округления. Это делает его незаменимым инструментом в различных областях, где требуется решение линейных систем.